REMARKS/ARGUMENTS

By this paper, Applicant responds to the Office Actions of March 24, 2005 and July 12, 2005 and respectfully requests reconsideration of the application. A Petition of Extension of Time extends the time to respond through September 24, 2005.

Claims 1-45 are now pending, a total of 45 claims. Claims 1, 9, 13, 16, 22, 24, 34, 36 and 40 are independent. Claims 1-33 are allowed. Claim 40 is now amended into independent form, with no amendment to scope, and is indicated allowable.

I. Interview of August 26, 2005

A telephone interview was conducted on August 26, 2005. No agreements were reached on ultimate issues. However, agreement on underlying issues was reached, and particular points of disagreement were identified. Specifics are set forth below in the context to which they are relevant.

II. Claim 36

Claim 36 is discussed in paragraph 3 of the Action of March 2005, and in comments to the Advisory Action of July 2005. Claim 36 recites as follows:

> 36. A computer, comprising:
>
> hardware designed to recognize a condition rising during execution of an instruction on a computer, in which the instruction is to affect the function to be performed by a second instruction;
>
> hardware and/or software designed to respond to the recognizing by setting a processor of the computer into single-step mode; and
>
> hardware and software designed to respond to execution of the second instruction by setting the computer out of single-step mode.

A. Claim 36 is Patentable on the Merits

A bit of background may be useful to address some of the issues raised in the interview.

The relationship between a debugger and a target program using the "industry standard processor such as the Pentium" is analogous to the relationship between an operating system and a user program. In particular, when a debugger is used to debug a target program, at any one time, either the debugger software is in execution, or the target program is in execution, **never** both. Just as an operating system and a user program can never be executing simultaneously on the same CPU, neither can a debugger and its target.

The analogy extends a bit further. In the operating system context, recall that during execution of a user program, certain events may occur that are detected by hardware, so that the processor can be switched from the user program to the operating system. Examples include divide-by-zero, page faults, and the like. When such an event occurs, the user program is suspended. Control is given to an operating system handler. When the handler has completed, control is returned to the user program. Note that the two programs are never in execution at the same time.

Analogously, a user of the debugger may specify certain conditions in the target program that are to cause a "break" into the debugger. Alpert's background describes such hardware support, for example, breakpoint registers and the like. Alpert's hardware debug features are not instructions themselves; rather, they operate somewhat analogously to page fault hardware: they detect various "break events" that arise during the execution of target programs. When such a "break event" occurs, the user program is suspended, and a debug "handler" goes into execution. Alpert '679, Fig. 3 ref. 330, col. 7, lines 36-55. When the user asks the debugger to resume execution of the user program, the debugger does so by moving the user program from "suspended" state to "ready to execute" state, and terminating the handler. Alpert '679, Fig. 3 ref 350, col. 8, lines 36-44. Thus, the debugger software and the target program being debugged are **never** in execution simultaneously. (That's why a user who is in the debugger perceives that the target program is in a "frozen" state – it can't execute until the debugger surrenders control back to the target program.)

The Examiner asked about the "debug hardware" mentioned at Alpert '679, col. 1, lines 15-16. The answer is straightforward: any "debug hardware" that is active during target program execution is not executable "instructions" *per se*, and the part of the debugger that is made of instructions does not operate during target program execution.

The Alpert '679 patent notes this mutual-exclusivity of instructions at several points, for example at col. 9, line 21-27, and lines 49-61, describing that the target program is suspended when the debugger begins, and vice-versa:

> As previously described, in response to each event the processor suspends execution of the current process, stores the execution environment of the suspended process in suspend instruction pointer register 148 and suspended status register 160, and begins execution of the appropriate handler. ...
>
> Upon completion of each handler executed in response to an event, the execution environment of the previously suspended process is restored--i.e., the contents of suspended status register 160 are copied into status register 150. As a result, enable bit 152 is restored to its state prior to the event--i.e., the state used by the suspended process. In addition, the state of mode indication 156 is restored to its state prior to the event. Thus, if the suspended process was executing in the user mode, restoring the execution environment of the suspended process causes the processor to switch to operating in the user mode by altering the state of mode indication 156. The execution of the suspended process is then resumed.

See also Alpert '679 Fig. 3, refs. 330 and 350, Fig. 5b, ref. 550 (noting that the target process is "suspended" during execution of the debugger and "resumed" when the debug handler completes), and col. 2, lines 28-45 (describing a number of conditions that limit the debugger to operating only at instruction boundaries of the target program, never simultaneously with execution of an instruction of the target program).

With that background, it is seen that the situations raised in the interview cannot possibly invoke the limitations of claim 36. Any "condition" recognized by Alpert's "debug hardware" affects only the very instruction currently executing, not a "second instruction" as recited in claim 36.

As noted in § II.B below, the Office Action and Advisory Action do not set forth any position in sufficient detail to permit a specific response.

B. As Procedural Matter, Neither Claim 34 Nor 36 Has Been Rejected

The sum total of the discussion of claim 36 in the March 2005 Action is as follows:

> 3. Alpert taught (claims 34, 36) recognizing an condition including exception condition, and in response, setting the processor into single-step mode; and taking single-step exception after executing the second instruction, and setting the processor out of single-step mode in industry standard processor such as the Pentium (e.g., see col. 1, line 13-col. 2, line 63). Also processor exceptions in processors that perform more than one task at a time such as the Pentium are well known to include conditions that affect another instruction such as writing to a location to read by the other instruction.

Merely setting out an inaccurate and partial paraphrase of the claim language, and designating a column-and-a-half of a reference, where that reference discusses two entirely separate embodiments, without identifying the relationship between claim terms and the reference, or how the two separate portions of the reference relate to each other, does not helpfully convey the basis for the Examiner's concern. It is also very unhelpful when portions of the claim language are entirely left out of the Office Action. Applicant cannot discern the Examiner's thoughts with respect to omitted claim language, or respond thereto.

The Advisory Action remarks as follows:

> [T]he prior art (Alpert) clearly teaches e.g., the situation of detection of a debug event, initiating a debugger, when the debugger is finished the processor is disclosed as resuming the original sequence of instruction, also the portions cited in the outstanding rejection disclosed that the system is put in step mode for performing the debug handler. Further the situation of a branch to another sequence of instruction (which may comprise one or plural instructions) the debugger is used in step mode so in response to completion of the second sequence of instruction in step mode the system returns (leaves step mode) to continue processing the original sequence of instructions.

This is no more helpful than the original Office Action. What instruction of Alpert corresponds to the first "instruction" of claim 36? For example, the Advisory Action does not even indicate whether the first "instruction" is part of the debugger, part of the target program, or part of the operating system. What does that instruction do? What "condition" is recognized by hardware during execution of that first instruction? What is the "second instruction?" How is the first instruction to "affect" the second? Where does the reference disclose "responding to the recognizing" by setting a processor into single-step mode? Until the Examiner sets to paper a reasonably complete and precise correspondence between the reference and the limitations recited in the claim, no detailed response is possible.[1]

During the interview, the Examiner observed that Alpert's debugger is inherently composed of instructions. Applicant agrees with that observation, but questions the relevance of

[1] 37 C.F.R. § 1.104(c)(2) sets the basic minima for any rejection over prior art. In the interview, it was agreed that these requirements had not been met. As a matter of administrative law, no rejection exists. Thus, no amendment is made in response to any rejection of any claim.

such a broad-brush statement. The claim recites two instructions with particular limitations, a particular condition, and particular interrelationships among them. The mere observation that the debugger has "instructions," with no identification of which instruction is relevant, is not helpful in conveying the Examiner's views on any limitation-by-limitation relationship between the claim and the reference.

In the interview, it was agreed that any future Office Action would precisely identify the relationship between every limitation in the claim and any reference relied upon, by both designating the portions of references "as nearly as practicable" and "explaining the pertinence," in the manner required by 37 C.F.R. § 1.104(c)(2). Applicant further requests that any reliance on "inherency" or "common knowledge" be accompanied by the "basis in fact and/or technical reasoning" or "specific factual findings predicted on sound technical and scientific reasoning," set forth explicitly as required by MPEP §§ 2112(IV) and 2144.03, and that even inherent components be identified "as nearly as practicable" and "as specific[ally] as possible" as required by 37 C.F.R. § 1.104(c)(2) and (d)(2).

III. Claim 34

Claim 34 recites limitations that are somewhat analogous to those recited in claim 36. For similar reasons, claim 34 is not rejected.

IV. Dependent claims

Dependent claims 35, 37-39, 44 and 45 are patentable with the independent claims discussed above. In addition, the dependent claims recite additional features that further distinguish the art.

V. Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. Enclosed is Petition for Extension of Time for three months. In the event that any further extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-29-000125BS.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: September 7, 2005

By: [illegible]

David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8757
(212) 728-9757 Fax